ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE

CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

Named Entity Extraction: l'approccio Machine Learning nella piattaforma GATE/ANNIE

Relatore:
Prof. Dario Maio

Candidato:
Andrea Zamberletti

Anno Accademico 2015/2016

Indice

Indice delle figure

Indice delle tabelle

Glossario degli acronimi

COR	Coreference Resolution
CREOLE	Collection of REusable Objects for Language Engineering
EE	Event Extraction
IDE	Integrated Development Environment
IE	Information Extraction
IR	Information Retrieval
JAPE	Java Annotation Pattern Engine
LHS	Left-Hand-Side
LR	Language Resource
ML	Machine Learning
MT	Machine Translation
MUC	Message Understanding Conference
NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
POS	Part Of Speech
PR	Processing Resource
RE	Relationship Extraction
RHS	Right-Hand-Side

Abstract

Scopo di questo testo è presentare l'attività di *Named Entity Extraction*, focalizzando l'attenzione sull'approccio *Machine Learning* e sulle tecniche implementate nella piattaforma GATE/ANNIE. È dapprima fornita una panoramica del settore *Natural Language Processing*, analizzandone le radici storiche, le problematiche principali e la complessità dovuta all'ambiguità del linguaggio umano. Successivamente è approfondito il task di *Information Extraction*, all'interno del quale si inserisce proprio l'attività di *Named Entity Extraction*. Si introduce quindi il tema *Machine Learning*, presentato prima in una visione più generale e poi inserito nel settore *Natural Language Programming*. Infine viene presentata la piattaforma GATE/ANNIE, descrivendone i vari componenti e illustrando un esempio di utilizzo per lo svolgimento di un'attività di estrazione di entità sfruttando algoritmi di *Machine Learning.*

1. Introduzione

Lo sviluppo vertiginoso di Internet e delle tecnologie digitali ha portato a una diffusione su larga scala di ogni tipo di contenuto testuale. Ogni giorno miliardi di persone si scambiano messaggi reciprocamente, esprimono opinioni sugli argomenti più svariati e pubblicano materiale di informazione o di intrattenimento. Tutti questi testi, memorizzati e scambiati sotto forma di dati digitali, possono esprimere diversi significati per chi li legge, ma sono poco più di sequenze di caratteri per un computer o un qualsiasi dispositivo elettronico che si occupa quotidianamente della loro creazione, conservazione e gestione. In un mondo in cui i dati costituiscono una risorsa sempre più importante, grazie alle crescenti potenzialità di elaborazione che la tecnologia offre, è inevitabile chiedersi quale impatto potrebbe avere la capacità di estrarre automaticamente informazioni a partire dall'oceano di risorse testuali attualmente non sfruttate.

Da questi presupposti si sviluppa la ricerca relativa a *Information Extraction*, il cui obiettivo è propriamente quello di sviluppare sistemi informatici in grado di estrapolare informazioni strutturate da documenti concepiti per l'interpretazione umana. L'idea di poter implementare sistemi dalle simili potenzialità rappresenta indubbiamente una sfida stimolante per un informatico.

All'interno del presente testo si vuole dunque cercare di osservare più da vicino le possibilità offerte dalla ricerca in questo ambito e i risultati già raggiunti. L'indagine viene effettuata dapprima attraverso un inquadramento ad ampio raggio del problema affrontato, analizzando il settore *Natural Language Processing*, all'interno del quale *Information Extraction* si inserisce. Successivamente l'analisi focalizza invece

l’attenzione su alcuni aspetti più specifici, prendendo come caso di studio il *task* di *Named Entity Recognition* e come software di riferimento la piattaforma GATE/ANNIE, una delle principali nell’ambito dell’elaborazione del linguaggio.

2. Natural Language Processing

Natural Language Processing (NLP) è un ambito dell'intelligenza artificiale che si occupa dei problemi legati all'analisi e all'elaborazione dei linguaggi naturali, ovvero degli strumenti comunicativi che caratterizzano l'interazione tra gli esseri umani. Al contrario dei linguaggi formali, generalmente utilizzati nell'interazione tra uomo e macchina, i linguaggi naturali non sono definiti da regole precise che ne permettano un'interpretazione univoca, ma presentano eccezioni, sinonimi e figure retoriche, oltre ad essere fortemente mutevoli in base al tempo e al luogo in cui vengono impiegati, fattori che li rendono soggetti ad ambiguità. Per questo la ricerca sul NLP è volta principalmente allo studio dei meccanismi intellettivi che consentono alle persone di risolvere tali ambiguità per decifrare i contenuti di una comunicazione, e di come essi possano essere riprodotti tramite algoritmi eseguibili dai calcolatori (Dale, 2010). Tali attività presentano una certa complessità e necessitano di diversi tipi di competenze che spaziano dalle scienze informatiche alla linguistica.

2.1 Breve storia di NLP

L'interesse nei confronti delle tematiche dell'elaborazione del linguaggio naturale è presente nell'uomo da ancora prima dell'esistenza di computer digitali. In particolare, nell'ambito di *Machine Translation* (MT), si può risalire fino al XVII secolo per trovare idee sulla meccanizzazione dei processi di traduzione, tramite la creazione di dizionari basati su codici numerici universali (Hutchins, Somers,1992).

Bisogna però arrivare al XX secolo per trovare i primi esperimenti significativi. Dopo un suo coinvolgimento in operazioni di decodificazione

di messaggi criptati durante la guerra, lo scienziato e matematico Warren Weaver cominciò a immaginare di poter usare le nuove tecnologie già utilizzate in ambito bellico per tradurre documenti da una lingua ad un'altra, ipotizzando di vedere la lingua stessa come un codice. Queste idee furono espresse dallo stesso Weaver in un memorandum del 1949, che generò un forte interesse nei confronti di MT. Negli anni successivi infatti vennero formati diversi gruppi di ricerca in America ed Europa in tale ambito e nel 1954 vi fu una prima dimostrazione pubblica delle potenzialità di tale disciplina con l'esperimento di Georgetown, sviluppato congiuntamente dall'Università di Georgetown e IBM, in cui furono tradotte automaticamente da un computer oltre sessanta frasi dal russo all'inglese. Sebbene il sistema fosse piuttosto semplice, la dimostrazione ebbe un discreto successo e generò ottimismo nei confronti dello sviluppo di tecnologie di MT.

Tuttavia la ricerca si rivelò molto più complessa del previsto e gli sviluppi furono scarsi negli anni successivi, tanto che nel 1966 l'ALPAC, un comitato di sette scienziati istituito dal governo degli Stati Uniti per valutare il progresso della ricerca in ambito di linguistica computazionale e MT, nello stilare il proprio rapporto sconsigliò di continuare ad investire in progetti di traduzione automatica, che al momento risultava più lenta, meno accurata e a costo doppio rispetto alla traduzione umana e non mostrava prospettive di miglioramento a breve termine. Ciò comportò un calo di investimenti nella ricerca, che subì pertanto un forte rallentamento negli anni successivi.

Una nuova svolta vi fu però verso la fine degli anni '80, quando furono introdotti algoritmi di *Machine Learning* nei sistemi di NLP, fino a quel momento basati solo su regole definite manualmente. Questo nuovo approccio ai problemi dell'elaborazione dei linguaggi naturali si dimostrò

particolarmente efficace e diede nuova linfa alla ricerca.

Con l'avvento di Internet e la diffusione di tecnologie digitali, infine, si sta assistendo a una presenza sempre maggiore di dati non strutturati che i sistemi informatici di ogni tipo devono gestire; questo rende ad oggi la ricerca su NLP di grande attualità ed interesse.

2.2 Task di Natural Language Processing

Le attività di NLP pongono diverse sfide dal punto di vista computazionale, data la complessità dell'oggetto di studio, il linguaggio umano. Tuttavia tale complessità è giustificata dalla varietà di possibili applicazioni che l'elaborazione del linguaggio naturale può trovare. Tra esse alcune delle principali sono:

- *Automatic Summarization*, produzione di un sommario degli argomenti trattati in un testo, quale un articolo o un saggio su un dato argomento;
- *Discourse Analysis*, analisi di un discorso a più livelli, anche considerando il contesto in cui è inserito e la relazione tra le varie parti che lo compongono;
- *Information Retrieval*, ramo che si occupa di gestire l'organizzazione e l'accesso a documenti conservati in memoria, così da poter selezionare in modo mirato gli oggetti contenenti informazioni rilevanti rispetto a specifiche *query* (è più legato al settore dell'informatica e dei database, ma si basa su tecniche di NLP);
- *Information Extraction*, estrapolazione di informazioni semantiche da dati non strutturati, quali documenti testuali;
- *Machine Translation*, traduzione automatica di testi da una lingua ad un'altra;
- *Natural Language User Interfaces*, interfacce per l'interazione tra

uomo e macchina in cui funzioni di controllo sono effettuate attraverso l'uso di strumenti linguistici; un tipico esempio è rappresentato da un motore di ricerca che non restituisce risultati in base alla presenza di *keyword* specifiche in un documento, ma è in grado di interpretare una richiesta espressa in linguaggio naturale e di fornire una risposta adeguata (ricerca semantica);

- *Optical Character Recognition*, ricostruzione di un testo tramite il riconoscimento dei caratteri che lo compongono in un'immagine;
- *Question Answering*, implementazione di sistemi in grado di dare risposte automatiche a domande formulate in linguaggio umano;
- *Sentiment Analysis*, il cui obiettivo consiste nel determinare la posizione dell'autore di un testo nei confronti dell'oggetto della discussione (che può essere un'idea, un prodotto, una persona, ecc.);
- *Speech Recognition*, rappresentazione testuale di una conversazione data in input sotto forma di file audio.

I task sopra elencati rappresentano una porzione dell'insieme dei campi applicativi di N*atural Language Analysis*, la parte di NLP che si occupa dell'analisi di testi o di conversazioni orali in linguaggio naturale e che è stata finora oggetto di maggiori ricerche. Tuttavia, un'area altrettanto importante e stimolante è costituita da *Natural Language Generation*, che si dedica alla produzione di contenuti testuali o vocali in forma facilmente comprensibile da utenti umani.

Nel capitolo successivo saranno approfonditi gli obiettivi e le sfide legati a *Information Extraction*, con focus sul problema *Named Entity Recognition*.

2.3 Conoscere la lingua

Si è precedentemente detto che gran parte delle difficoltà incontrabili nell'elaborazione del linguaggio naturale derivano dalla necessità di

risolvere ambiguità proprie del linguaggio stesso. Per farlo è quindi necessario comprendere come esso sia strutturato e come diversi aspetti di esso concorrano a renderne possibile un'interpretazione piuttosto che un'altra.

Il campo di ricerca che si occupa di definire e comprendere le caratteristiche del linguaggio è la linguistica, che è composta principalmente da sei sotto-discipline, corrispondenti in linea di massima ai livelli che compongono un sistema lingua:

- fonetica e fonologia, riguardanti lo studio dei suoni di una lingua;
- morfologia, lo studio dei morfemi, ovvero delle strutture interne di una parola;
- sintassi, la quale si occupa della struttura delle frasi in base alle relazioni tra le diverse parole che le compongono;
- semantica, riguardante il significato delle parole e delle frasi nel complesso;
- pragmatica, si occupa di come il contesto influisca sull'interpretazione dei significati di una frase;
- analisi del discorso, lo studio di unità linguistiche lunghe più di un enunciato.

Basandosi su questa divisione si può ricondurre quasi tutte le attività di NLP a tentativi di risolvere ambiguità ad uno di questi livelli (Jurafsky, Martin, 2009). Per esempio in un'operazione di S*peech Recognition* si potrebbe incontrare un'omofonia, caratteristica di segni grafici differenti che rappresentano lo stesso suono (in italiano un caso è costituito dalla c- di cuore e la q- di quadro); mentre analizzando la frase <<*Rapina in banca con rivoltella da centomila euro*>> si presenterebbe un'ambiguità sintattica, in quanto il complemento di stima "da centomila euro" potrebbe riferirsi alla rapina come alla rivoltella.

2.4 Processo di Natural Language Analysis

Proprio questa corrispondenza tra i diversi livelli della lingua e le ambiguità che le tecniche di NLP devono risolvere fa sì che un processo di analisi di linguaggio naturale sia composto da una sequenza di passi che riflette in parte la struttura definita precedentemente:

1. pre-elaborazione del testo,
2. analisi lessicale,
3. analisi sintattica,
4. analisi semantica,
5. analisi pragmatica.

Si forniscono di seguito alcuni dettagli relativi a ciascuna fase, per chiarirne le funzioni e le competenze.

2.4.1 Pre-elaborazione del testo

Si tratta della prima fase del processo di analisi, nella quale un file testuale "grezzo", formato essenzialmente da una sequenza di bit, viene convertito in una sequenza ben definita di unità linguisticamente rilevanti: caratteri, parole e frasi (Palmer, 2010). La fase di pre-elaborazione del testo si può a sua volta suddividere nei passi appresso elencati:

a. *Document Triage*, processo di conversione di un insieme di file digitali in un corpus di documenti testuali ben definiti, attraverso il riconoscimento della codifica dei caratteri, l'identificazione della lingua del testo e l'esclusione di elementi non rilevanti nell'analisi testuale quali *header*, immagini, ecc.

b. *Tokenization*, passo in cui la sequenza di caratteri in input è divisa in *token*, che possono essere parole, numeri o simboli di punteggiatura. Questo compito può sembrare relativamente

semplice, tuttavia può risultare non così banale soprattutto per quelle lingue dove non è presente spaziatura tra le parole, come il cinese o il tailandese.

c. *Sentence segmentation*, identificazione dei confini delle diverse frasi che compongono il documento.

2.4.2 Analisi lessicale

Questa parte del processo si focalizza sulla struttura delle parole, e si basa quindi su alcuni concetti della morfologia. In particolare è necessario introdurre due termini usati in tale ambito per entrare nel dettaglio di questa fase: il tema e il lemma. Il tema è la radice di una parola, ottenibile rimuovendo da una sua forma flessa (ad esempio la coniugazione di un verbo o il plurale di un sostantivo) la desinenza. Il lemma è invece la forma canonica della parola, ovvero quella che viene convenzionalmente scelta per rappresentarne tutte le forme flesse; in italiano ad esempio il lemma di un verbo è la sua coniugazione all'infinito presente, di un aggettivo il singolare maschile. La fase di analisi lessicale ha il compito di individuare tema e lemma di ogni parola attraverso due differenti operazioni, rispettivamente *stemming* (da *stem*, inglese per tema) e *lemmatization*. Queste informazioni saranno poi utilizzate nelle successive fasi di analisi; risulta infatti molto meno dispendioso, in termini di memoria, mantenere regole basate sulle parti che compongono una parola e su come le loro combinazioni vadano a formare determinate forme flesse, piuttosto che gestire ogni parola come un elemento atomico all'interno di un enorme inventario (Dale, 2010). Questo è facilmente riscontrabile pensando ad esempio alla struttura di un dizionario. Al suo interno sono infatti elencati solo i lemmi, a cui ci si può rifare anche per l'interpretazione del significato delle forme non canoniche; difficilmente si potrebbe

immaginare un volume contenente tutte le forme flesse di tutte le parole, per le dimensioni e per la difficoltà di utilizzo che questo risulterebbe avere.

2.4.3 Analisi sintattica

Per comprendere il significato di una frase non è sufficiente conoscere il significato delle parole che la compongono, ma è essenziale sapere anche come esse siano in relazione tra loro. L'analisi sintattica ha come obiettivo quello di assegnare un ruolo ad ogni parte della frase in modo da ricostruirne la struttura, che viene generalmente rappresentata come albero sintattico.

Un albero sintattico è composto da più livelli di raggruppamento delle parole. Le foglie rappresentano il risultato dell'operazione di *Part-Of-Speech tagging* (POS *tagging*) ed hanno cardinalità pari al numero di parole della frase, in quanto ad ognuna è associata una parte del discorso (verbo, sostantivo, aggettivo, articolo, ecc.). I rami intermedi risultano invece dal passo di *chunking*, che raggruppa tra loro più parti del discorso sintatticamente correlate. Alla cima vi è la frase stessa.

La Figura 1 mostra un semplice esempio di albero sintattico in cui vengono utilizzati i *tag*: S - *sentence* (per la frase), NP - *noun phrase* (frase nominale), VP - *verbal phrase* (frase verbale), V - *verb* (per i verbi), N - *noun* (per i sostantivi), Det (per gli articoli).

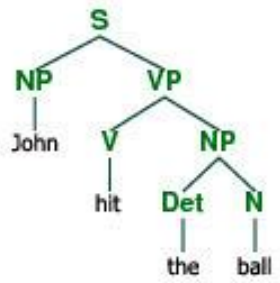


Figura 1. Esempio di albero sintattico della frase inglese "John ha colpito la palla"

2.4.4 Analisi semantica

La fase di analisi semantica sfrutta le informazioni ottenute dai precedenti passi di analisi relativamente ai significati delle singole parole, e alle relazioni tra esse, per interpretare il significato della frase nel complesso. Nonostante questo compito sia centrale nella maggior parte dei task di NLP, quali *Machine Translation*, *Information Extraction* e *Text Summarization*, ad oggi gran parte della ricerca legata all'elaborazione del linguaggio naturale si è concentrata sulla morfologia e sulla sintassi, basando l'intero processo di analisi su un'operazione di riconoscimento di parole o di pattern. Questo approccio di elaborazione al livello delle parole, tuttavia, difficilmente può risultare in una reale comprensione del significato di un testo. Per questo motivo la ricerca sul NLP dovrà gradualmente incentrarsi sull'analisi semantica dell'intera frase per raggiungere un livello adeguato di comprensione del linguaggio umano (Cambria, White, 2014).

Alcune operazioni riconducibili alla fase di analisi semantica sono:

- *Finding Collocation*: ovvero l'identificazione e interpretazione di quelle sequenze di parole (definite appunto *collocation*) che assumono un proprio significato non deducibile da quello delle singole parti;
- *Word Sense Disambiguation*: operazione che mira a determinare

quale dei vari significati attribuibili ad una singola parola sia quello migliore nel contesto della frase in cui la parola è inserita.

2.4.5 Analisi pragmatica

In questa fase si pone attenzione sull'intenzione dell'autore (o del parlante se si analizza una conversazione orale) più che sul significato specifico di una frase. Il linguaggio umano infatti si basa non solo sulle proprie caratteristiche morfologiche, sintattiche e semantiche, ma anche su conoscenze esterne, legate al contesto in cui una frase è inserita. Per poter correttamente interpretare il messaggio di una comunicazione potrebbe infatti essere necessario, ad esempio, avere:

- conoscenza del ruolo e dello status degli interlocutori,
- collocazione spazio-temporale della situazione,
- conoscenza dell'argomento trattato.

Anche questo tipo di analisi è ancora poco approfondito in letteratura, soprattutto a causa delle grandi difficoltà che presenta.

2.5 Principali tool

Con l'avanzare della ricerca nel settore, anche la disponibilità di piattaforme per l'esecuzione di task di NLP cresce sempre di più; di seguito si elencano le principali.

- GATE, software open-source sviluppato dall'Università di Sheffield a partire dal 1995, è composta da una suite di strumenti per la risoluzione di diversi problemi legati a NLP. Una delle componenti principali di GATE è ANNIE, un sistema che offre funzionalità di base per *Information Extraction*. Nel capitolo 5 si entrerà nel dettaglio della sua architettura, concentrandosi in particolare sulle possibilità che esso offre per utilizzare un approccio

machine learning nell'esecuzione di alcuni task di NLP;

- Stanford CoreNLP: piattaforma sviluppata dal Natural Language Processing Group dell'Università di Stanford, distribuita con licenza GPL, che include diversi tool per l'analisi di documenti in linguaggio naturale;
- Natural Language Toolkit: è la principale piattaforma per lavorare con dati in linguaggio naturale in Python. Come le altre piattaforme offre diversi tool per l'esecuzione di vari task del NLP;
- Apache OpenNLP: set di strumenti sviluppato da Apache Software Foundation, basato su *Machine Learning*;
- Apache UIMA: anch'esso sviluppato da Apache Software Foundation, è un'implementazione dell'interfaccia *Unstructured Information Management Architecture* per l'analisi di dati non strutturati.

3. Information Extraction e Named Entity Recognition

3.1 Information Extraction

Come messo in luce precedentemente, l'esplosione del Web e l'avvento dei *social network* rappresentano una grande opportunità per il settore del NLP, offrendo una base importante di documenti e un campo fertile per lo sviluppo di diversi task dell'elaborazione del linguaggio naturale. Basti pensare che, se dalla nascita di Internet al 2003, anno di apparizione di alcuni dei principali *social network* (MySpace, LinkedIn, Facebook), erano presenti sul Web solo alcune dozzine di exabyte di informazioni, oggi la stessa quantità di contenuti è creata settimanalmente e gran parte di essa è costituita da dati non strutturati (Cambria, White, 2014). Si può affermare, di conseguenza, che volendo reperire una certa informazione, il problema non stia tanto nella possibilità che essa esista e sia raggiungibile, quasi sempre verificata, quanto sulla sua individuazione ed estrazione all'interno di un oceano di informazioni non strutturate. Lo stesso Tim Berners-Lee, inventore del *World Wide Web,* definisce la rete Internet di oggi come *Web of documents*, confrontandolo con un maggiormente auspicabile modello di *Web of data*, che potrebbe essere immaginato come "la rete delle cose del mondo, descritte sul Web da dati" (Berners-Lee, Bizer, Heath, 2009). In questo scenario, essenziale è proprio l'obiettivo di *Information Extraction* (IE), che cerca di ricavare dati strutturati da documenti non strutturati o semi-strutturati. Lo scopo di IE è quindi quello di estrarre le sole informazioni semanticamente rilevanti in un testo, per renderle facilmente consultabili da un essere umano e manipolabili da un computer.

3.2 Sviluppo di IE

L'interesse verso IE si può far risalire alla fine degli anni '70, con primi rilevanti progetti ad uso commerciale che sono emersi già a metà del decennio successivo (Cowie, Wilks, 1996); si può citare in particolare il sistema JASPER, costruito dal Carnegie Group per Reuters con lo scopo di fornire notizie finanziarie *real-time* agli operatori di borsa. Successivamente la ricerca continuò e ottenne progressi grazie soprattutto all'organizzazione di progetti come le *Message Understanding Conferences* (MUC), eventi organizzati per incentivare lo sviluppo di nuove tecniche per IE dalla Defense Advanced Research Projects Agency (DARPA), agenzia del Dipartimento di Difesa degli Stati Uniti d'America che si occupa di sviluppare tecnologie emergenti per uso militare. Le MUC consistevano in sfide di IE a cui la comunità scientifica partecipante era invitata a prendere parte, mettendo insieme le proprie tecnologie per risolvere determinate problematiche proposte (Piskorski, Yangarber, 2012). Da queste conferenze sorsero gran parte dei task di IE oggi riconosciuti e delle principali tecnologie associate. La Tabella 1 schematizza le diverse edizioni di MUC svolte.

Conferenza	Anno	Tipo di fonti	Dominio
MUC-1	1987	Rapporti militari	Operazioni navali
MUC-2	1989	Rapporti militari	Operazioni navali
MUC-3	1991	Agenzie stampa	Attività terroristiche in America Latina
MUC-4	1992	Agenzie stampa	Attività terroristiche in America Latina
MUC-5	1993	Agenzie stampa	*Joint venture* corporative, produzione di mircoelettronica
MUC-6	1995	Agenzie stampa	Negoziazione di vertenze sindacali e successione manageriale
MUC-7	1997	Agenzie stampa	Incidenti aerei e lanci missilistici

Tabella 1. Riepilogo delle edizioni di MUC e delle caratteristiche dei testi elaborati

In seguito, altri eventi e programmi, tra cui il programma di *Automated Content Extraction* (ACE), portarono avanti il lavoro iniziato con le MUC, contribuendo ad un progressivo avanzamento delle tecnologie oggetto della ricerca su IE e mettendo in luce nuove sfide da affrontare. Tra queste attualmente stanno assumendo particolare importanza gli studi su tecniche di IE indipendenti dalla lingua del testo o specifiche per lingue diverse dall'inglese, sulla quale si è basato il lavoro di ricerca negli ultimi decenni. Inoltre si sta introducendo nella ricerca il tema dell'estrazione di informazioni da fonti multiple, particolarmente importante per operazioni di *fact-checking* o per confrontare informazioni, anche contraddittorie, derivanti da origini diverse o che abbiano subito un'evoluzione nel tempo (Piskorski, Yangarber, 2012).

3.3 Task di Information Extraction

Un sistema di IE non ha come scopo la comprensione totale di un testo, ma una mirata ricerca ed estrazione di specifiche informazioni. Si possono infatti definire le attività di IE come caratterizzate dalle due seguenti proprietà (Jurafsky, Martin, 2009):

1. il tipo di conoscenza che si desidera estrarre può essere descritto attraverso un *template* fissato e relativamente semplice, composto da più *slot* (o attributi) che vengono riempiti da un sistema di IE, durante l'elaborazione, con una stringa presa dal testo stesso, con un valore scelto da una lista predefinita o con un riferimento ad un template elaborato precedentemente;
2. solo una piccola parte delle informazioni presenti in un testo è rilevante per riempire un singolo *template*.

Rientrano in tale definizione i seguenti *task*, in cui si può suddividere IE:

- *Named Entity Recognition* (NER), consiste nell'identificazione

e nella classificazione di predefiniti tipi di entità all'interno del testo. Esempi classici di entità riconoscibili in un documento sono organizzazioni, persone, luoghi, ecc.

- *Coreference Resolution* (COR), la quale individua multiple menzioni di una stessa entità nel testo.
- *Template Element construction* (TE), operazione che, anche grazie ai risultati di COR, aggiunge informazioni descrittive ad un'entità.
- *Relationship Extraction* (RE), individuazione e classificazione di relazioni predefinite esistenti tra entità.
- *Event Extraction* (EE), attività che ha l'obiettivo di identificare eventi nel documento ed estrarre informazioni strutturate relative ad essi. Idealmente il risultato di EE individua "chi ha fatto cosa a chi, dove, quando, attraverso quali metodi (o strumenti) e perchè" (Piskorski, Yangarber, 2012). Questo *task* prevede in genere l'estrazione di diverse entità e relazioni che le colleghino tra loro.

Si consideri, per chiarire le competenze di ciascun task, la seguente porzione di testo:

> *"Il 28 giugno 2016 il gruppo cinese Suning Holdings Group, di proprietà dell'imprenditore Zhang Jindong, acquista il 68,55% dell'Inter, divenendo così l'azionista di maggioranza del club milanese."*

Ipotizzando di eseguire su questa frase i cinque *task* sopra definiti, i risultati ottenuti sarebbero simili a quelli illustrati di seguito.
Risultati di NER: "28 giugno 2016" - DATA

“Suning Holdings Company” - ORGANIZZAZIONE

“Zhang Jindong” - PERSONA

“Inter” - ORGANIZZAZIONE

“68,55%” - PERCENTUALE

“club milanese” - ORGANIZZAZIONE

Risultati di COR: “club milanese” → “Inter”

Risultati di TE: DATA-1:

Valore: “2016-06-28”

ORGANIZZAZIONE-1:

Nome: “Suning Holdings Group”

Nazionalità: “cinese”

PERSONA-1:

Nome: “Zhang”

Cognome: “Jindong”

Professione: “imprenditore”

PERCENTUALE-1:

Valore: “68,55”

ORGANIZZAZIONE-2:

Nome: “Inter”

Tipo: “club”

Nazionalità: “italiana”

Sede: “Milano”

Risultati di RE: ProprietàDi(“Zhang Jindong” , “Suning Holdings Group”)

AcquistatoDa(“Inter” , “Suning Holdings Group”)

Risultati di EE: EVENTO-ACQUISIZIONE-1:

Data: DATA-1

Acquirente: ORGANIZZAZIONE-1

Oggetto: ORGANIZZAZIONE-2

Quota: PERCENTUALE-1

La rappresentazione sopra riportata dei risultati ottenibili non solo schematizza le differenze tra i tipi di informazioni e di elaborazioni effettuate su esse per ciascun task, ma evidenzia come ogni attività definisca, per ciascuna informazione ricercata, un *template* preciso che specifica quali dati siano da considerare rilevanti e quali si possano invece ignorare. La struttura dei risultati, inoltre, risulta facilmente riconducibile ai record di un database, mettendo in luce come l'obiettivo primario delle attività di IE sia di avere dati facilmente manipolabili da macchine.

3.4 Named Entity Recognition

Questo lavoro vuole concentrare l'attenzione sull'analisi delle attuali tecniche disponibili per l'esecuzione del task di NER. Questa scelta è dovuta a diversi motivi, da un lato si tratta di una delle attività su cui si è maggiormente focalizzata la ricerca in ambito di IE e costituisce la base per gli altri *subtask*, dall'altro presenta una vasta quantità di possibili applicazioni in campi diversi della scienza contemporanea.

3.4.1 Problematiche legate a Named Entity Recognition

L'estrazione di entità all'interno di un testo implica due principali compiti:

- segmentazione, fase in cui devono essere individuati i confini di un'entità;
- classificazione, assegnazione ai segmenti individuati di

informazioni semantiche rilevanti;

Questo duplice obiettivo computazionale, all'interno dell'attività di estrazione delle entità, fa sì che ci si riferisca ad essa anche come *Named Entity Recognition and Classification* (NERC).

I possibili approcci ai due problemi possono essere di tre tipi:

- Basato su *lookup list*, quindi sul riconoscimento di parti del testo all'interno di liste predefinite suddivise per categorie. È un sistema semplice, ma che richiede liste estremamente complesse ed estese di possibili entità, difficili da stilare e gestire.
- *Rule-based*, in cui il testo viene analizzato secondo un insieme di regole che determinano la presenza di entità all'interno di un testo e ne definiscono le relative informazioni semantiche. Le regole utilizzate fanno uso, generalmente, di informazioni di tipo ortografico e sintattico. La complessità del linguaggio rende molto difficile la stesura di un set efficace di regole se il contesto è abbastanza complesso, ma con un buon lavoro di studio e di *testing* è possibile raggiungere un elevato livello di precisione.
- Statistico, generalmente attraverso l'uso di tecniche di *machine learning*. Questo approccio consiste nell'identificare pattern nei dati forniti in input per determinare in modo probabilistico dove potrebbe essere presente un'entità. Le tecniche relative all'approccio statistico per il NER saranno esplorate nel capitolo successivo.

3.4.2 Valutazione delle performance di un sistema di Named Entity Recognition

Data la complessità del problema NER e la presenza di svariate tecniche per la sua risoluzione è importante poter valutare le prestazioni di un sistema, per comprendere se possa rappresentare un approccio valido.

A tale scopo è necessario confrontare i risultati ottenuti dal sistema in analisi con un *gold standard*, ovvero con l'insieme corretto delle entità presenti sullo stesso documento (ottenuto in genere da un'elaborazione di etichettatura manuale). Il confronto viene rappresentato attraverso tre principali valori:

- *precision* (P), rapporto tra il numero di entità corrette e il totale delle entità estratte;
- *recall* (R), rapporto tra il numero di entità corrette estratte e il totale delle entità presenti nel *gold standard*;
- *F-score* (F): un indicatore calcolato come media pesata di *precision* e *recall*.

F-score è generalizzabile attraverso un parametro β che determina la rilevanza di P rispetto a R, secondo la formula:

$$F_\beta = (1 + \beta^2) \cdot \frac{1}{(\beta^2 \cdot P) + R}$$

L'F-score più utilizzato è F_1 che dà lo stesso peso a R e P e rappresenta la media armonica tra i due.

4. Machine Learning

Un sistema informatico risolve problemi computazionali tramite algoritmi, ovvero sequenze di operazioni eseguite sui dati forniti in input per ottenere determinati risultati. Tuttavia non sempre è definibile un algoritmo valido universalmente per eseguire un certo compito, a causa dell'estrema variabilità dei dati in input e della mancanza di un modello che ne definisca l'elaborazione per quel compito. Si pensi ad esempio ad un'operazione di determinazione dello *spam* in una casella di posta elettronica; in questo caso si è a conoscenza della struttura dei dati in input ed è noto il tipo di risultato da ottenere, ma non è definito uno schema che mappi le informazioni fornite dai dati in modo da restituire una classificazione appropriata di ciascuna mail. Quello che potrebbe essere disponibile è tuttavia un insieme di dati già mappati, dove quindi si possa associare un determinato input ad uno specifico risultato dell'operazione. Nell'esempio introdotto precedentemente, questo significherebbe avere a disposizione un set di mail correttamente marcate come *spam* o come messaggio desiderato (ci si potrebbe basare ad esempio sulle segnalazioni effettuate manualmente dagli utenti). Partendo da questa base, si può cercare di insegnare ad una macchina come risolvere il problema in oggetto, ovvero come estrarre automaticamente l'algoritmo, analizzando i risultati disponibili (Alpaydin, 2010). Questo tipo di processo fa parte dei metodi computazionali studiati da *Machine Learning* (ML), una delle principali aree dell'Intelligenza Artificiale.

In generale, si può definire ML come il settore che si occupa dello studio e della costruzione di algoritmi tramite i quali una macchina sia in grado di evolvere il proprio comportamento basandosi su dati empirici. Caratteristica propria dei processi di apprendimento automatico oggetto del

ML è quindi l'induzione, in quanto il modello alla base di un algoritmo consiste in una generalizzazione ottenuta da un insieme di dati osservati, piuttosto che in un insieme di regole definite da applicare ai casi specifici. Inoltre dalla definizione di ML emerge un'altra importante proprietà degli algoritmi studiati, quale l'evoluzione, il progressivo miglioramento conseguente dall'analisi di una maggiore quantità di dati. Questo aspetto è ben formalizzato nella definizione di apprendimento automatico fornita dall'informatico Tom Mitchell: "Si dice che un programma informatico apprende dall'esperienza E rispetto a una classe di task T e una scala di misurazione delle performance P, se la sua performance nei task di T, misurata secondo P, migliora con l'esperienza E" (1997).

4.1 Definizione di un problema di Machine Learning

Un problema di ML può essere definito come l'individuazione, dato un insieme **T** *(training set)* composto dalle coppie $(\boldsymbol{x}_i,y_i)$ dei dati in input $\boldsymbol{x}_i$ e dei corrispondenti valori di output y_i, di un modello che associa ad ogni elemento $\boldsymbol{x}$, detto **istanza**, un valore y, tale che applicato ad un $\boldsymbol{x}_i$ restituisca y_i per ogni $(\boldsymbol{x}_i,y_i) \in \mathbf{T}$. Un'istanza $\boldsymbol{x}$ è un vettore $(x_{i(1)},x_{i(2)},\ldots,x_{i(d)})$ i cui elementi $x_{i(1)},x_{i(2)},\ldots,x_{i(d)}$ sono le relative ***feature***, o attributi.

Esistono diversi tipi di problemi trattabili con algoritmi di ML, appartenenti ad un ampio spettro di campi di applicazione, ma possono essere raggruppati in base ad alcune caratteristiche.

Una distinzione fondamentale è effettuata tra problemi affrontabili con algoritmi di tipo:

- *supervised learning*, in cui il sistema viene istruito tramite esempi, ovvero dati in input dei quali è specificato il corretto valore di output;
- *unsupervised learning*, se al contrario i dati in input non sono

etichettati per dare un riferimento nell'impostazione del modello, in questo caso spetta all'algoritmo stesso trovare il modello dei dati attraverso il riconoscimento di pattern;

- *reinforcement learning*, altro caso in cui non è fornito un set di esempi, in questo tipo di algoritmi tuttavia viene fornito un *feedback* sulla correttezza dei risultati ottenuti dai vari passi di elaborazione.

Tecniche *supervised* risultano chiaramente più efficaci, tuttavia non è sempre possibile avere a disposizione una sufficiente quantità di dati pre-annotati, situazione che rende necessario l'utilizzo di algoritmi di tipo *unsupervised.*

Un'ulteriore suddivisione è invece specificata dal tipo di valore di output dell'algoritmo. Si definiscono in particolare le due tipologie di problema appresso specificate.

- *Classification*, se il valore di output appartiene a un insieme limitato di valori, detti classi. Si parla di *clustering* se il valore di output è sì ristretto a un numero limitato di classi, ma queste non sono note a priori; si tratta di conseguenza di un problema affrontato in modalità *unsupervised*, il cui obiettivo è quello di raggruppare istanze appartenenti alla stessa classe, senza però poterne determinare il tipo.
- *Regression*, è il caso in cui il valore di output dell'algoritmo sia di tipo numerico.

4.2 Machine Learning e Natural Language Processing

L'introduzione di tecniche di *Machine Learning* in NLP, avvenuta intorno alla fine degli anni '80, ha rappresentato una vera e propria rivoluzione nel settore (Johnson, 2009). Il precedente approccio, basato su set di regole definite a priori, trovava infatti grosse difficoltà nella creazione di metodi efficaci, se non con elevati sforzi di programmazione e per domini

estremamente limitati, a causa della natura estremamente irregolare e mutevole della lingua. La possibilità di applicare modelli statistici all'analisi del linguaggio ha invece reso alcuni task maggiormente generalizzabili ed efficienti, fornendo una serie di vantaggi:

- maggiore robustezza rispetto a input atipico o erroneo;
- possibilità di focalizzarsi automaticamente sui casi più comuni, non sempre facilmente individuabili da un programmatore;
- capacità di migliorarsi attraverso l'analisi di quantità maggiori di dati di apprendimento.

Questi e altri motivi hanno permesso una rilevante diffusione di algoritmi di ML in diversi tipi di task di NLP.

4.3 Approccio machine learning a Named Entity Recognition

Come illustrato nel più sopra il problema NER implica prima di tutto un'attività di segmentazione. Un tipico approccio, basato su ML, al problema così definito consiste nel vedere NER come un problema di *sequence labeling* (o *sequence tagging*), attività di *pattern recognition* per la categorizzazione, attraverso l'assegnamento di etichette, degli elementi che compongono una sequenza. Nel caso del riconoscimento di entità le etichette rispettano lo schema IOB (*Inside Outside Beginning*), secondo il quale ogni *token* può essere contrassegnato come I, se è interno ad una NE, come O se non ne fa parte e come B se è all'inizio di un segmento di testo rappresentante un'entità (Hobbs, Riloff, 2010). In quest'ottica anche il task di segmentazione è visibile come una forma di classificazione, tuttavia gli algoritmi *sequence labeling* considerano un'istanza non solo rispetto ai propri attributi ma anche in relazione ad elementi adiacenti. Alcuni esempi di modelli su cui si basano gli algoritmi utilizzati per questo tipo di task sono: *Hidden Markov Models*, *Maximum Entropy Markov Models*,

Conditional Random Fields e *Support Vector Machines* (Hobbs, Riloff, 2010; Erdogan, 2010).

5. GATE

GATE (*General Architecture for Text Engineering*) è un'infrastruttura per lo sviluppo e il *deployment* di componenti software per l'elaborazione di linguaggio umano (Cunningham, 2014), il cui sviluppo ha avuto inizio presso l'Università di Sheffield nel 1995 e che è ancora oggi largamente usata per ogni tipo di attività computazionale legata al linguaggio naturale.
GATE offre software gratuito e *open-source* e può essere pensato come un'architettura per *Language Engineering* (LE). Per LE si intende la costruzione di sistemi di NLP i cui costi di produzione e output sono predicibili e misurabili.

5.1 La famiglia di prodotti di GATE

GATE non è costituito solo da un insieme di strumenti software, ma è un progetto ad ampio raggio, che comprende diverse soluzioni dedicate ad attività di elaborazione del linguaggio naturale. I prodotti facenti parte di tale progetto sono:

- GATE Embedded, *framework object-oriented* sviluppato in Java che fornisce gli strumenti software alla base del progetto GATE;
- GATE Developer, IDE per lo sviluppo di applicazioni per NLP, basato su GATE Embedded, ma con strumenti che ne facilitano l'uso quali un'interfaccia grafica e strumenti di *debug*;
- GATE Cloud, un sistema distribuito e parallelo che permette di eseguire applicazioni GATE *on demand* su hardware altamente prestativo;
- GATE Teamware, un ambiente per l'annotazione collaborativa di documenti;
- GATE Mìmir, piattaforma per l'implementazione di funzionalità di

indicizzazione e ricerca su testi, annotazioni e schemi e metadati semantici.

- GATE Wiki, un Wiki/CMS, che permette modifiche collaborative, anche off-line in modo asincrono, usato principalmente come risorsa per operazioni di *testing*;
- The GATE Process, un servizio di tutoraggio dedicato ad organizzazioni che vogliono avvalersi di tecniche di elaborazione del linguaggio;
- GATE Training and Certification, una serie di corsi, online o intensivi svolti in contesti universitari, per l'apprendimento delle tecniche legate all'utilizzo della piattaforma GATE.

Nel corso del capitolo saranno analizzate in maggiore dettaglio alcune delle componenti di GATE e saranno mostrate alcune delle funzioni di GATE Developer. Per maggiori informazioni riguardo l'intera infrastruttura GATE e i singoli prodotti che ne fanno parte, si rimanda invece alla documentazione online (disponibile all'URL https://gate.ac.uk/sale/tao/split.html).

5.2 GATE Embedded

GATE Embedded consiste in una libreria di classi che implementano le funzionalità alla base dell'elaborazione di documenti testuali e costituisce, di fatto, il nucleo della piattaforma GATE. La libreria è inoltre ottimizzata per essere inclusa dagli sviluppatori nelle proprie applicazioni per eseguire task di NLP.

5.2.1 Architettura di GATE

La libreria Gate Embedded è strutturata in modo da rispettare i concetti di modularità e riusabilità delle proprie classi. In particolare, l'intera

architettura di GATE è basata su componenti, porzioni di codice indipendenti definite da interfacce che possono essere implementate in modi diversi, secondo il modello dei *Java Beans*. Tali componenti possono essere suddivisi in tre categorie:

- LanguageResources (LR), rappresentano i modelli di corpora, documenti e annotazioni su testi;
- ProcessingResources (PR), componenti di tipo prevalentemente algoritmico che elaborano risorse testuali, restituendo specifiche informazioni;
- VisualResources (VR), modelli che definiscono la visualizzazione e la modifica di altre risorse e sono usate per la costruzione di interfacce grafiche.

L'insieme delle risorse integrate in GATE è detto CREOLE (*a Collection of REusable Objects for Language Engineering*). Tutte le risorse di CREOLE sono rappresentate da una classe Java e da un file di configurazione XML presenti allo stesso URL, tramite il quale possono essere individuate dal *framework* di GATE, che si occupa di caricarle e di gestirne i processi e l'utilizzo di memoria.

La Figura 2 mostra uno schema della libreria GATE Embedded e delle sue componenti.

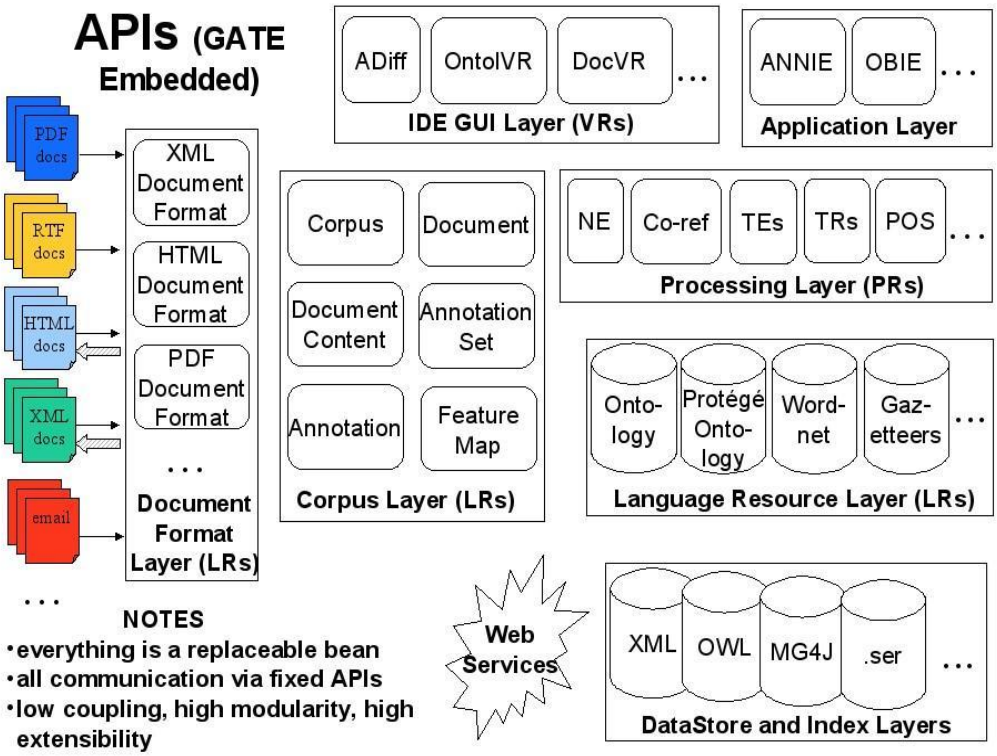


Figura 2. architettura di GATE Embedded

5.2.2 LanguageResources (LR)

Le LR in GATE possono essere documenti, corpora (set di documenti), o annotazioni.

Un'annotazione è una forma di metadato associato ad una specifica porzione di un documento, definita da due "puntatori" che indicano l'inizio e la fine della stringa di riferimento all'interno del contenuto; ogni annotazione è inoltre caratterizzata da un id numerico, da un tipo e da un insieme di *feature* che ne descrivono informazioni aggiuntive. La Tabella 2 mostra una rappresentazione delle annotazioni risultanti da un'operazione di *tokenization* con *POS-tagging* sul testo "Luigi mangia la mela."; inizio e fine della stringa annotata sono indicate nella tabella come offset rispetto al primo carattere del testo e l'unica *feature* presente specifica il tipo di POS del *token* annotato (i tipi di POS utilizzati sono NP per i nomi propri, VB per i verbi, DT per gli articoli e NN per i nomi comuni).

ID	Type	SpanStart	SpanEnd	Features
1	token	0	4	pos=NP
2	token	6	11	pos=VP
3	token	13	14	pos=DT
4	token	16	19	pos=NN

Tabella 2. Esempio di rappresentazione di annotazioni risultanti da tokenization e POS-tagging

Le annotazioni associate a parti di un documento vengono rappresentate come parti del documento stesso all'interno di *annotation set.* Un documento ha sempre un *annotation set* di default, al quale possono esserne aggiunti altri indefinitamente.

I formati di documenti supportati da GATE automaticamente sono: file di testo, HTML, SGML, XML, RTF, alcuni file PDF, e*mail*, alcuni formati di Microsoft Office e di OpenOffice; inoltre è fornita l'opportunità di integrare la possibilità di utilizzo di altri formati particolari tramite l'uso di *plugin* (ad esempio per file JSON contenti *tweet* o per *markup* MediaWiki, utilizzati da Wikipedia e altri siti).

GATE è in grado di gestire file testuali strutturati o semi-strutturati associando al documento le informazioni in esso contenute sotto forma di annotazioni. In particolare al momento dell'importazione di un documento, GATE ricerca all'interno del contenuto l'eventuale presenza di *tag*, che, qualora individuati, saranno rappresentati da annotazioni create ad hoc il cui tipo corrisponderà al nome del *tag* e le cui *feature* saranno estratte dagli attributi dello stesso.

Un *tag* espresso nella forma:

```
<aTagName attrib1="value1" attrib2="value2" attrib3="value3">
     A piece of text
</aTagName>
```

conseguirà quindi nella creazione di un'annotazione come segue.

```
annotation.type = "aTagName";
annotation.fm = {
      attrib1=value1;
      atrtrib2=value2;
      attrib3=value3
};
annotation.start = startNode;
annotation.end = endNode;
```

5.2.3 Processing Resources

Le PR costituiscono le componenti di elaborazione dei task di NLP eseguibili in GATE. Possono essere utilizzate singolarmente, in quanto moduli indipendenti, o essere combinate insieme ad altre PR per formare delle applicazioni. L'esecuzione di più PR all'interno di un'applicazione può essere di due tipi:

- *pipeline* semplice, in cui PR all'interno di un insieme ordinato sono eseguite una alla volta;
- *pipeline* su corpus, specifico per gruppi di PR eseguiti su documenti e corpora, che esegue sequenzialmente tutte le PR che compongono l'applicazione su un documento prima di aprire il successivo all'interno del corpus, sul quale verranno ripetute le stesse operazioni.

5.3 JAPE

JAPE (*Java Annotation Patterns Engine*) è un linguaggio di *pattern-matching* disponibile in GATE, che permette di riconoscere *regular expression* su annotazioni in un documento. Nonostante la struttura secondo la quale sono organizzate le annotazioni in GATE sia un grafo, quindi più complessa rispetto a quelle normalmente gestibili da linguaggi regolari, spesso i dati descritti dai grafi di annotazioni possono essere

considerati come semplici sequenze e permettono un riconoscimento di tipo deterministico. In altri casi, tuttavia, la maggiore complessità del dato in input può rendere non sufficiente la potenza computazionale di un automa a stati finiti e implicare la necessità di un'elaborazione alternativa da parte di JAPE.

Una JAPE *grammar*, in particolare, consiste in una serie di fasi, specificate secondo un preciso ordine logico, in modo tale che annotazioni temporanee generate da una prima fase, possano essere fornite come input per una successiva. Ogni fase è costituita da set di regole composte da *pattern*, il cui riconoscimento innesca l'esecuzione di azioni specificate. Una regola è formata da due parti: una *Left-Hand-Side* (LHS), che contiene la descrizione dei *pattern* da riconoscere sulle annotazioni, e una *Right-Hand-Side* (RHS), che definisce invece le istruzioni da eseguire su annotazioni sulle quali i *pattern* siano stati riconosciuti. Una JAPE *grammar* presenta inoltre per ogni fase un *header* che ne specifica il nome, l'input e alcune opzioni aggiuntive relative alle modalità di applicazione delle regole.

Si fornisce di seguito un semplice esempio di JAPE *grammar*.

```
Phase: UrlPre
Input:  Token SpaceToken
Options: control = appelt

Rule: Urlpre
(
(({Token.string == "http"} |
{Token.string == "ftp"})
{Token.string == ":"}
{Token.string == "/"}
{Token.string == "/"}) |
({Token.string == "www"}
{Token.string == "."})
):urlpre
-->
:urlpre.UrlPre = {rule = "UrlPre"}
```

Nell'esempio rappresentato è definita una fase "UrlPre" che prende come

input annotazioni risultanti da un'operazione di *tokenization*. "UrlPre" è costituita da una singola regola che cerca di riconoscere se la stringa del token rappresenta la parte iniziale di un URL *web* tramite il *pattern* definito dalla LHS (codice che precede l'operatore `-->`) e, in caso di *match*, aggiunge un'annotazione UrlPre tramite la RHS (codice che segue l'operatore `-->`).

JAPE rappresenta il principale strumento per lo sviluppo di task di NLP con approccio *rule-based*. Non rientrando tra gli obiettivi del presente testo quello di approfondire tale approccio, non viene fornita una descrizione ulteriormente dettagliata del linguaggio JAPE e di relative funzionalità più avanzate; per un eventuale approfondimento si rimanda alla documentazione presente sul sito di GATE (https://gate.ac.uk/sale/tao/splitch8.html).

5.4 ANNIE

GATE fu inizialmente sviluppato nel contesto della ricerca su IE (Cunningham, 2014), per questo una parte fondamentale della propria piattaforma è dedicata proprio a tale task di NLP. Si tratta di ANNIE (*A Nearly New Information Extraction system*), un sistema composto da una serie di moduli, basati su algoritmi a stati finiti e istruzioni JAPE, che permettono di elaborare annotazioni utilizzabili per vari task di IE. La Figura 3 mostra una rappresentazione delle componenti di ANNIE, tra cui le principali saranno analizzate in dettaglio di seguito.

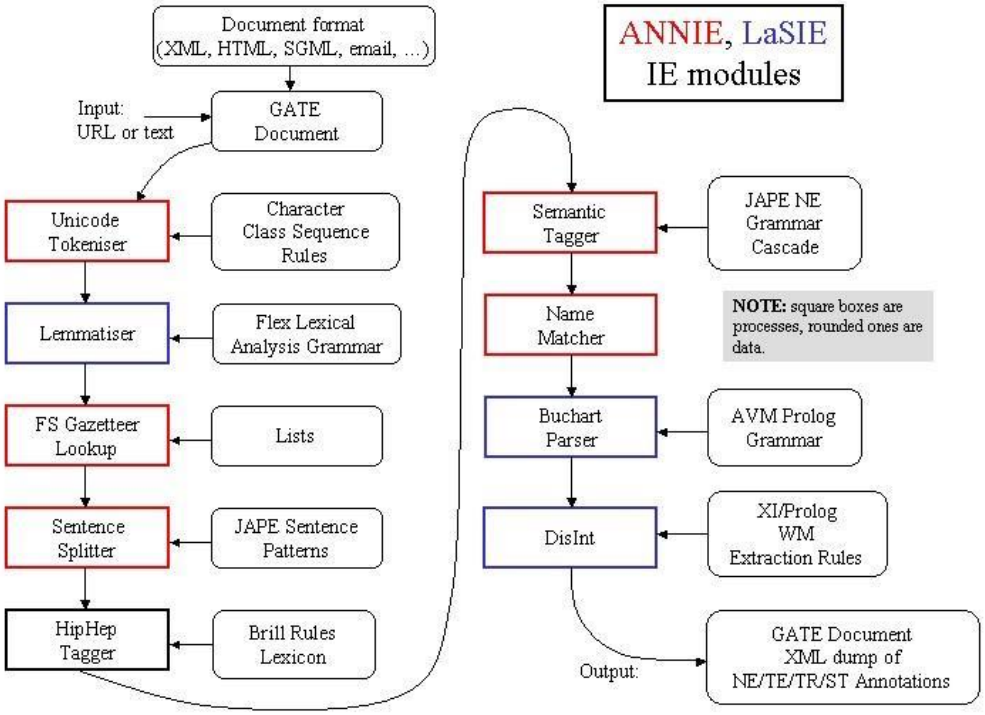


Figura 3. Rappresentazione dei moduli che compongono ANNIE

5.4.1 Document Reset

Document Reset PR è una risorsa che consente di riportare il documento allo stato originale, rimuovendo annotazioni aggiunte in seguito ad elaborazioni precedenti. È configurabile grazie ad una serie di parametri che consentono di specificare se rimuovere tutte le annotazioni o solo alcune. È generalmente eseguita all'inizio di ogni applicazioni in ANNIE per ottenere un risultato non alterato da altre annotazioni.

5.4.2 Tokeniser

Questa PR può essere utilizzata per suddividere il contenuto di un documento in *token* (parole, numeri e simboli di punteggiatura). Le annotazioni risultanti dall'esecuzione di un *tokeniser* su un documento possono essere di due tipi, Token e SpaceToken, e hanno associate le *feature*:

- *string*, il valore stringa del *token* a cui è associata l'annotazione*;*
- *length*, il numero di caratteri di cui è composta la stringa;
- *kind*, attributo che distingue diversi tipi di *token.*

I possibili tipi di *token* definiti dall'attributo *kind* sono riportati di seguito.

Per annotazioni di tipo Token:

- *Word*, definito come un qualsiasi insieme di lettere maiuscole o minuscole adiacenti tra loro (può contenere anche il simbolo "-"). Le annotazioni con attributo *kind* uguale a *word* hanno un'ulteriore *feature* di nome *orth* che può assumere i valori *upperInitial*, *allCaps*, *lowerCase* o *mixedCaps* in base alla distribuzione di lettere maiuscole e minuscole all'interno del *token*.
- *Number*, definito come una qualsiasi combinazione di cifre consecutive.
- *Symbol*, rappresenta un qualsiasi numero di simboli contigui. Contiene una *feature* aggiuntiva di nome *symbolkind* che può essere uguale a *currency*, se il simbolo contenuto corrisponde a una valuta, o *other* altrimenti.
- *Punctuation*, rappresenta sempre un unico simbolo di punteggiatura. Può avere un attributo *position*, con valore *startpunct* se si tratta di un simbolo di punteggiatura di apertura (ad esempio una parentesi aperta) o con valore *endpunct* se contiene invece un simbolo di chiusura (ad esempio una parentesi chiusa).

Per annotazioni di tipo SpaceToken:

- *space*, se contiene puri spazi di separazione;
- *control*, se contiene caratteri di controllo (ad esempio il carattere di nuova riga).

La Figura 4 mostra la visualizzazione di annotazioni risultanti dall'esecuzione del *tokeniser* in GATE Developer. Nella parte superiore

sono rappresentati graficamente i *token* all'interno del testo, in azzurro quelli con annotazione di tipo Token e in verde quelli di tipo SpaceToken; nella parte inferiore invece viene fornita una rappresentazione tabellare delle informazioni associate ad ogni annotazione.

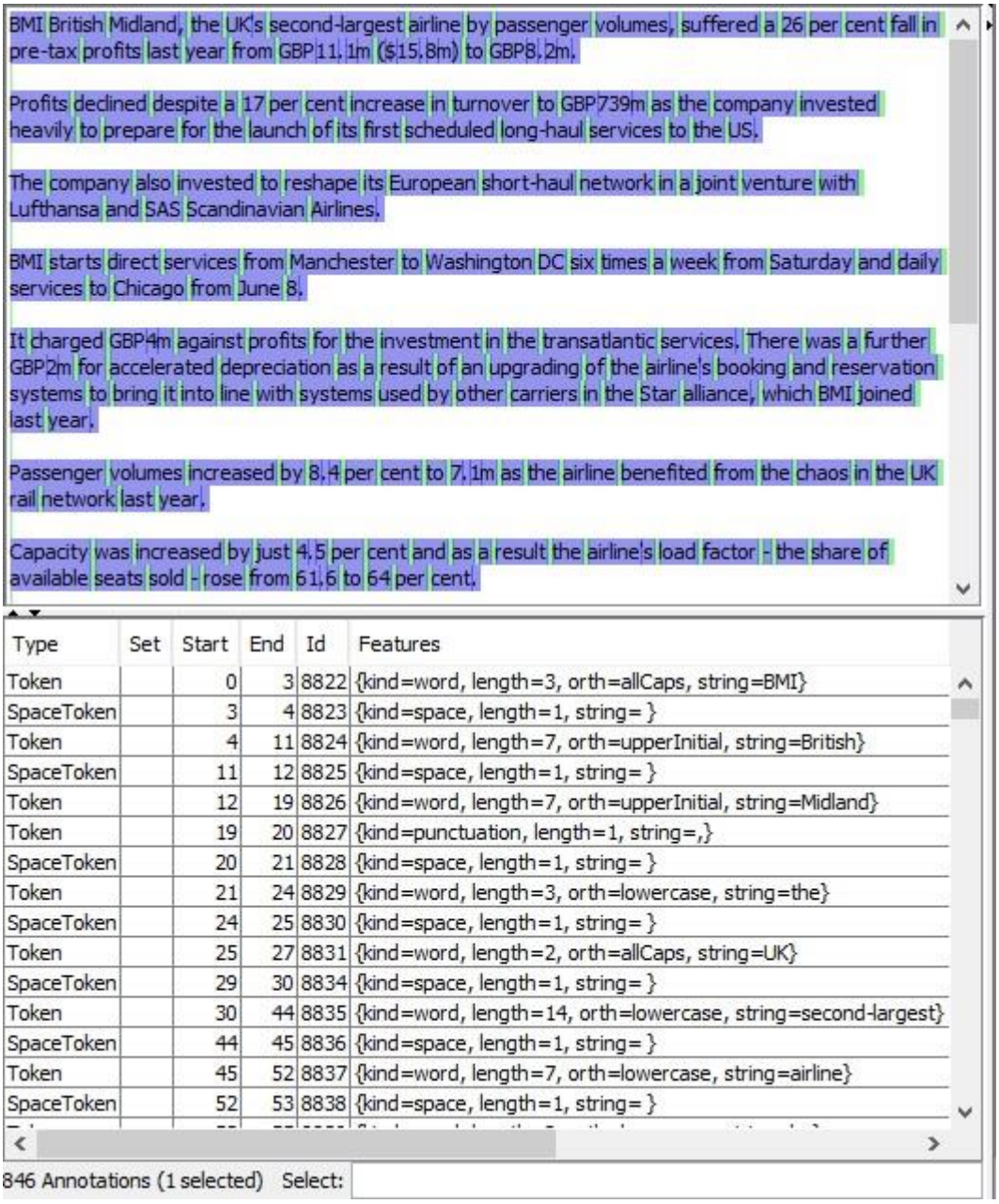


Figura 4. Esempio di annotazioni ottenute dall'esecuzione di un tokenizer in GATE Developer

Il tokeniser offerto di default da ANNIE è ANNIE English Tokeniser, che si compone di un normale *tokenizer* e di un *JAPE transducer*, utilizzato per ottimizzare i risultati in base alle caratteristiche della lingua inglese. Per

eseguire attività di *tokenization* su testi in altre lingue è disponibile anche un *tokenizer* generico, il GATE Unicode Tokeniser. La genericità di quest'ultimo strumento non lo rende estremamente adatto ad alcune lingue specifiche, ma può essere utilizzato come base per sviluppare un *tokenizer* ottimizzato (per alcune lingue ne esistono già alcuni reperibili online che permettono di ottenere buoni risultati).

5.4.3 ANNIE Gazetteer

Il ruolo di questa PR consiste nell'identificare entità all'interno del documento attraverso il confronto con delle *lookup list.* Una *lookup list* utilizzata dal Gazetteer consiste in un file di testo contenente una voce per riga; ciascun file raggruppa un insieme di nomi (di città, organizzazioni, mesi, ecc.). Una lista di valute potrebbe, ad esempio, contenere una porzione simile a questa:

```
Ecu
European Currency Units
FFr
Fr
German mark
German marks
New Taiwan dollar
New Taiwan dollars
NT dollar
NT dollars
```

L'ANNIE Gazetteer utilizza un file *lists.def* per gestire l'accesso alle liste. All'interno di questo file ogni lista è rappresentata nella forma *list_filename.lst:major_type:minor_type,* dove *list_fileame.lst* è il nome del file contenente la lista, mentre *major* e *minor type* sono due diverse classificazioni del tipo di entità da associare alle voci contenute nella lista, rispettivamente meno e più specifica. Ogni parte di testo che sia individuata all'interno di una lista è annotata dal Gazetteer con tipo Lookup e *feature majorType* e *minorType.* La forma precedentemente esposta può essere

inoltre integrata con una quarta colonna per l'indicazione della lingua dei termini contenuti nella lista e con una quinta che specifichi un tipo personalizzato di annotazione.

Le seguenti righe rappresentano un esempio di possibili voci del file *lists.def*.

```
currency_prefix.lst:currency_unit:pre_amount
currency_unit.lst:currency_unit:post_amount
date.lst:date:specific
day.lst:date:day
```

La PR ANNIE Gazetteer è configurabile attraverso la modifica diretta delle *lookup list* o tramite una serie di parametri che permettono di specificare opzioni relative al tipo di *match* che deve essere considerato (ad esempio se deve essere *case sensitive*).

5.4.4 Sentence Splitter

Il *sentence splitter* è una risorsa che si occupa di dividere il contenuto del documento in segmenti corrispondenti alle diverse frasi del testo. L'esecuzione di tale PR comporta la creazione di due tipi di annotazioni:

- Sentence, associata ad ogni periodo individuato;
- Split, associata ad ogni parte di testo corrispondente alla terminazione di una frase (ad esempio un punto). Le annotazioni di tipo Split hanno una *feature* di nome *kind* che assume valore *internal* per tutte le combinazioni di punti interrogativi ed esclamativi e per ogni sequenza costituita da un numero di punti compreso tra 1 e 4, *external* per caratteri di nuova riga.

Oltre alla PR fornita da ANNIE, ANNIE Sentence Splitter, è disponibile in GATE anche la risorsa RegEx Sentence Splitter, basata su *regular expression,* sviluppata per sopperire ad alcune lacune in fatto di *performance* mostrate dalla componente offerta da ANNIE, soprattutto in caso di input irregolari.

5.4.5 ANNIE POS Tagger

Questa risorsa di ANNIE permette di eseguire un'operazione di *POS-tagging* (descritta al Paragrafo 2.5.3) sui *token* che compongono un testo. Il risultato consiste nell'aggiunta di un attributo *category* ad ogni annotazione di tipo Token; il valore assunto da tale *feature* corrisponde a una parte del discorso della lingua del testo ed è rappresentato da un codice costituito da lettere e simboli. La lista completa dei codici utilizzati dalla PR è consultabile all'interno della documentazione di GATE (https://gate.ac.uk/sale/tao/splitap7.html).

5.4.6 ANNIE NE Transducer

L'ANNIE Named Entity Transducer è un annotatore semantico basato sul linguaggio JAPE, che si occupa di analizzare le annotazioni elaborate nelle fasi precedenti per cercare di identificare entità all'interno di un documento e restituirle sotto forma di nuove annotazioni.
I possibili tipi di entità, e le relative *feature*, prodotte da ANNIE nell'esecuzione di questa risorsa corrispondono ai tipi di entità definiti originariamente dalle MUC, e sono le seguenti:

- Person
    - *gender: male, female*
- Location
    - *locType: region, airport, city, country, county, province, other*
- Organization
    - *orgType: company, department, government, newspaper, team, other*
- Money
- Percent

- Date
    - *kind: date, time, dateTime*
- Address
    - *kind: email, url, phone, postcode, complete, ip, other*
- Identifier
- Unknown

5.4.7 OrthoMatcher

OrthoMatcher è la PR di ANNIE che fornisce la possibilità di individuare relazioni di coreferenza tra le entità annotate dal NE Transducer. Una coreferenza è presente quando due divere stringhe all'interno del testo rappresentano la stessa entità. Ad esempio nella frase "i vicini chiameranno la polizia, se a loro non sta bene il volume della musica" vi è una coreferenza tra il pronome "loro" e il soggetto della prima proposizione, "i vicini", a cui si riferisce.

OrthoMatcher non è in grado di trovare nuove entità all'interno del testo, ma può assegnare un tipo ad un'entità precedentemente annotata come Unknown (di tipo sconosciuto), utilizzando l'annotazione dell'entità riconosciuta come coreferenza. Le regole utilizzate per determinare la presenza di coreferenza tra due entità, sono applicate solo se le due relative annotazioni sono dello stesso tipo, o se una delle due è di tipo Unknown, al fine di evitare la riclassificazione di una *named entity* correttamente identificata. Il risultato del riconoscimento di una coreferenza consiste nell'aggiunta alle annotazioni delle due entità di una *feature* il cui valore è un puntatore all'altra entità.

5.5 Machine Learning in GATE

In accordo con i recenti sviluppi in ambito di NLP, anche GATE prevede

l'utilizzo di tecniche di ML. In particolare GATE implementa algoritmi di *supervised* ML per l'esecuzione di attività di *classification*, *regression* e *sequence tagging.*

Come visto nel Capitolo 4 (Paragrafo 4.1) un algoritmo di ML analizza delle istanze, in base a pattern individuati sulle relative *feature*, per assegnare a ciascuna un valore di output. In GATE possiamo vedere questo processo in base ai dati analizzati dalle PR che lo compongono: le annotazioni. Le annotazioni in GATE rappresentano le istanze che vengono valutate da un algoritmo di ML in base agli attributi che le definiscono. I valori di output possono essere invece il tipo di una nuova annotazione associata ad un'istanza o il valore di un attributo della stessa.

GATE fornisce la possibilità di eseguire algoritmi di ML attraverso tre differenti *plugin*:

- Learning Framework, il più recente *plugin* per l'utilizzo di tecniche di ML in GATE, è costituito da un *set* di risorse per l'esecuzione di diversi tipi di *task*;
- Batch Learning PR, nucleo del Learning *plugin*, è la versione precedentemente supportata, ed era mirata all'utilizzo di algoritmi per attività di *classification*, *sequence tagging* e *relation extraction*;
- Machine Learning PR, all'interno del Machine_Learning *plugin*, è la più vecchia delle soluzioni ML all'interno di GATE.

5.5.1 Learning Framework

Learning Framework è il *plugin* per *Machine Learning* attualmente supportato ed offre un numero di algoritmi maggiore rispetto alle precedenti versioni. In particolare si basa sulle librerie Mallet (*MAchine Learning for LanguagE Toolkit*, un pacchetto di strumenti JAVA per NLP statistico che comprende algoritmi di classificazione e *sequence tagging*) e

LibSVM (*a Library for Support Vector Machines*, libreria che implementa algoritmi per la classificazione a vettori di supporto), di cui supporta gran parte degli algoritmi. Inoltre permette l'integrazione di un numero sempre crescente di librerie per ML attraverso l'uso di *wrapper*, che permettono l'utilizzo di software esterno anche se distribuito con una licenza differente rispetto a quella di Learning Framework (di tipo *Lesser General Public License*) o sviluppato in un differente linguaggio.

Learning Framework è strutturato per offrire tre tipi di attività: *classification*, *regression* e *sequence tagging*, chiamato *chunking* all'interno di Learning Framework. Essendo tutti task eseguiti in modalità *supervised*, ognuno è implementato attraverso due PR, una di *training* e una di applicazione. Le PR di *training* permettono di generare un modello tramite il riconoscimento di pattern su un corpus pre-annotato. Gli attributi delle annotazioni che devono essere analizzati in questa fase sono definiti da un file XML fornito in input. Il modello risultante dalla fase di *training* è poi caricato dalle PR applicative per l'esecuzione dell'operazione sul corpus che si vuole elaborare. Per le attività di *classification* e *regression* sono presenti anche delle PR per la valutazione, che permettono di eseguire le operazioni riservando una parte dei documenti di *training* per un confronto con i risultati ottenuti.

Il *plugin* offre inoltre una PR per l'esportazione di dati in diversi formati, in modo da poter usare i propri dati anche al di fuori di GATE.

Nel paragrafo seguente verrà illustrato più nel dettaglio un esempio di utilizzo del Learning Framework all'interno di GATE Developer per l'esecuzione di alcune operazioni di NER.

5.6 Esempio di Named Entity Recognition con approccio Machine Learning in GATE Developer

Per rendere più chiare le modalità di utilizzo di tecniche di ML nell'esecuzione di un'operazione di NER, si fornisce un esempio svolto utilizzando GATE Developer in combinazione con ANNIE e Learning Framework.

5.6.1 GATE Developer

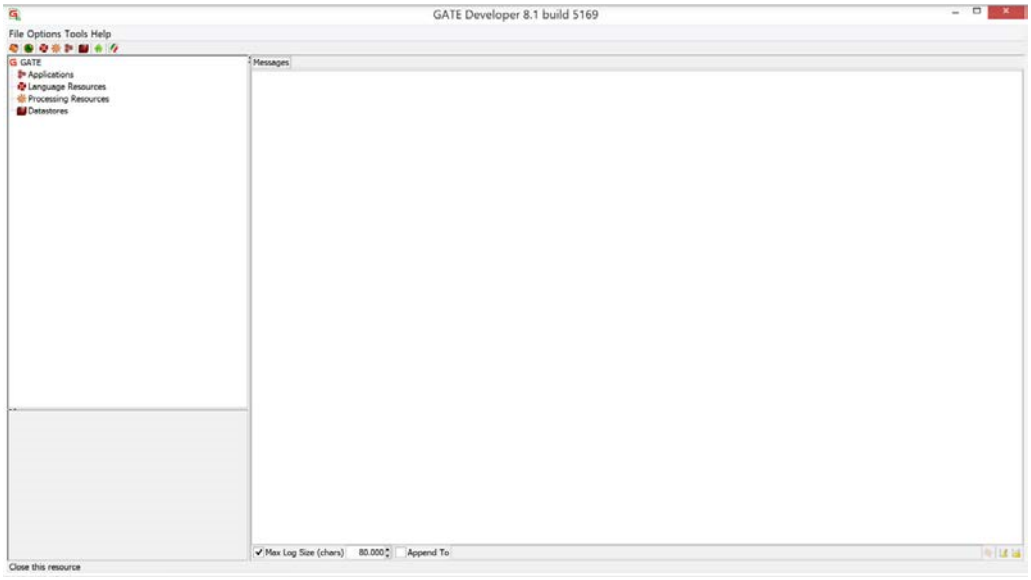


Figura 5. Interfaccia grafica di GATE Developer

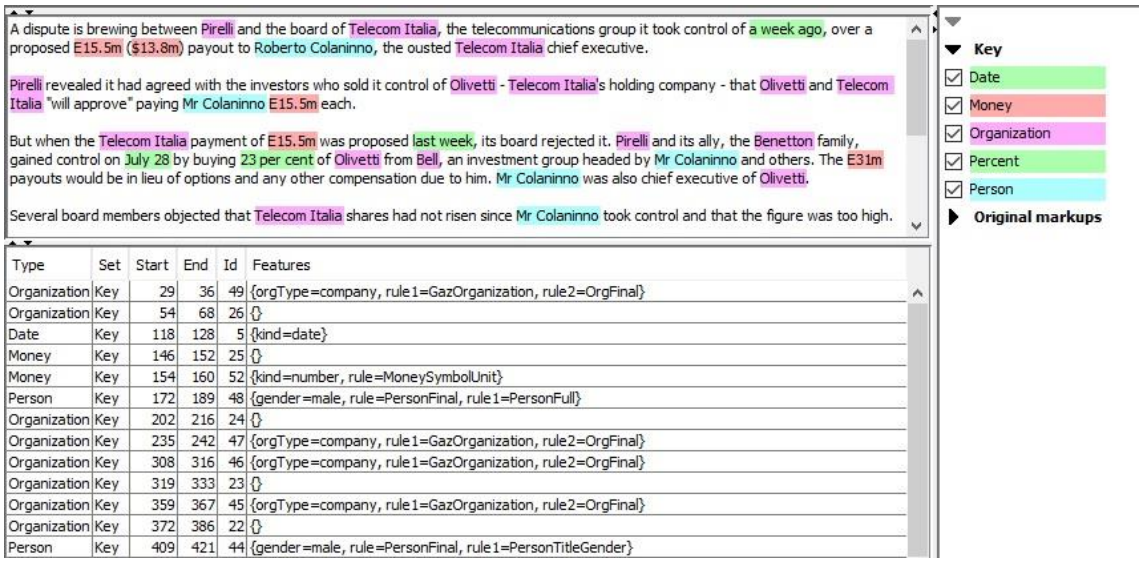
GATE Developer è l'interfaccia grafica di GATE e permette di utilizzare in modo intuitivo le risorse della piattaforma per la creazione di applicazioni eseguibili all'interno dell'IDE stesso o esportabili per essere integrate in software esterni.

La GUI di GATE Developer, mostrata in Figura 5, è formata da due parti principali, il *resources tree* e il *resource viewer*, più alcune parti utilizzabili per funzioni di controllo, quali menu e barre dei messaggi. Il *resources tree* è posizionato a sinistra della finestra di GATE Developer e rappresenta le varie risorse caricate, tra cui ci sono le LR, le PR e le applicazioni; da qui si

può accedere a ciascuna singola risorsa, che sarà aperta nel *resource viewer*. Il *resource viewer* è il pannello centrale della finestra di GATE Developer e permette di visualizzare e interagire con le risorse caricate. In particolare, su una LR aperta permette di visualizzare e modificare i contenuti del corpus o del documento selezionato, comprese le annotazioni e il testo in esso contenuti; su un'applicazione consente invece la selezione delle PR da eseguire e la loro configurazione attraverso l'impostazione di parametri *run-time.*

5.6.2 Descrizione dei corpora utilizzati per il test

Per il test effettuato sono stati utilizzati due differenti corpora: uno per la fase di *training* e uno per il test vero e proprio. Entrambi i corpora sono formati da documenti (46 il primo e 47 il secondo) contenenti brevi testi a carattere prevalentemente finanziario, a cui è stato associato un *annotation set*, chiamato "Key", che racchiude annotazioni di entità di tipo Date, Location, Money, Organization, Percent e Person. Tali annotazioni saranno usate durante la fase di *training* per la creazione del modello di elaborazione dell'applicazione e successivamente alla fase di test come confronto per la valutazione dei risultati. in Figura 6 è mostrato un esempio di documento appartenente a uno dei corpora con le annotazioni associate.



Type	Set	Start	End	Id	Features
Organization	Key	29	36	49	{orgType=company, rule1=GazOrganization, rule2=OrgFinal}
Organization	Key	54	68	26	{}
Date	Key	118	128	5	{kind=date}
Money	Key	146	152	25	{}
Money	Key	154	160	52	{kind=number, rule=MoneySymbolUnit}
Person	Key	172	189	48	{gender=male, rule=PersonFinal, rule1=PersonFull}
Organization	Key	202	216	24	{}
Organization	Key	235	242	47	{orgType=company, rule1=GazOrganization, rule2=OrgFinal}
Organization	Key	308	316	46	{orgType=company, rule1=GazOrganization, rule2=OrgFinal}
Organization	Key	319	333	23	{}
Organization	Key	359	367	45	{orgType=company, rule1=GazOrganization, rule2=OrgFinal}
Organization	Key	372	386	22	{}
Person	Key	409	421	44	{gender=male, rule=PersonFinal, rule1=PersonTitleGender}

Figura 6 . Esempio di documento annotato contenuto nei corpora utilizzati

5.6.3 Fase di training

Come anticipato nei paragrafi precedenti, trattandosi di un'operazione eseguita con algoritmi di *supervised* ML, il test si è composto di due differenti fasi, una di *training* e una di test, ciascuna delle quali ha previsto l'esecuzione di un'applicazione ANNIE su un corpus.

Per la fase di training sono state utilizzate in sequenza le seguenti PR:

1. Document Reset, PR eseguita all'inizio di ogni applicazione in GATE. Scopo di Document Reset è quello di eliminare annotazioni pre-esistenti che potrebbero interferire con l'elaborazione successiva. Pertanto per poter utilizzare le annotazioni contenute nel *set* "Key" per le fasi di elaborazione successive (in particolare da parte della Training Chunking PR del Language Framework), è stato necessario aggiungere l'*annotation set* nella lista delle annotazioni da mantenere attraverso il parametro *setsToKeep*.
2. ANNIE English Tokeniser, risorsa che si occupa della creazione delle annotazioni di tipo Token e delle relative *feature*. Le annotazioni risultanti dall'esecuzione di questa PR costituiranno le

istanze per l'algoritmo di *chunking*.

3. ANNIE Sentence Splitter, PR che genera annotazioni di tipo Sentence; queste ultime saranno utili per definire il contesto all'interno del quale sarà sensato, per l'algoritmo di *sequence tagging,* ricercare pattern utili a definire la segmentazione delle entità ricercate.
4. ANNIE POS Tagger, aggiunge ai *token* informazioni di tipo sintattico, che saranno inserite tra gli attributi analizzati dall'algoritmo di ML.
5. Annotation Set Transfer, utilizzata per trasferire le annotazioni presenti nel set "Key" all'interno del *default annotation set*. Qui è infatti dove ANNIE inserisce le proprie annotazioni, che Learning Framework si aspetta di trovare insieme alle annotazioni delle entità; sarebbe stato possibile anche trasferire le annotazioni di ANNIE nel set "Key", ma si preferisce non modificare le annotazioni originali. Per lo stesso motivo è importante impostare il parametro *copyAnnotations* a *true*, in modo che la PR non sposti le annotazioni ma le copi solo; infine è necessario definire il parametro *inputAsName* con il nome del set da cui copiare le annotazioni, in questo caso "Key".
6. LF_TrainChunking, il vero e proprio cuore della fase di *training*. Questa PR si occuperà di eseguire un algoritmo di *sequence tagging* sulle annotazioni ottenute dalle precedenti risorse, per restituire il modello che sarà utilizzato nella fase di test per l'estrazione delle entità. Per un corretto funzionamento della risorsa è necessario impostare una serie di parametri *run-time*, illustrati in Figura 7 e descritti brevemente di seguito.

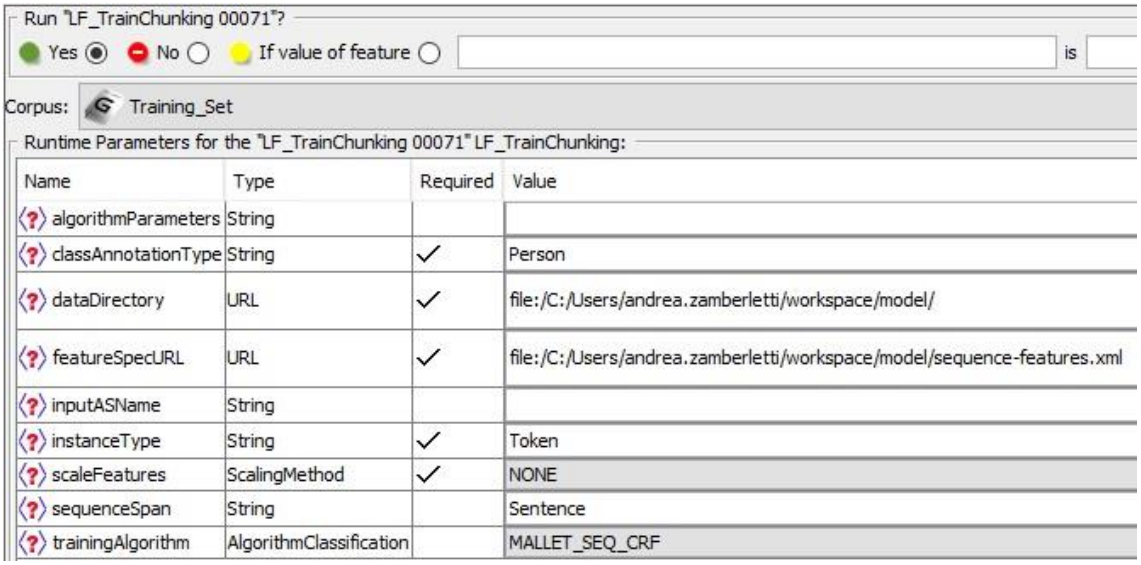


Figura 7. Parametri run-time per la configurazione della Training Chunking PR

Il primo parametro da definire è *classAnnotationType*, che rappresenta il tipo di entità di cui vogliamo trovare le sequenze all'interno del testo (nell'esempio mostrato sono state ricercate entità di tipo Person). L'URL fornito a *dataDirectory* serve a definire la posizione in cui verrà salvato il modello restituito dall'esecuzione dell'applicazione, mentre quello di *featureSpecURL* serve alla PR per localizzare il file di configurazione delle *feature*, necessario per interpretare la struttura degli attributi delle istanze analizzate (il codice del file utilizzato per il test è mostrato in Figura 8). Attraverso il parametro *instanceType* viene specificato il tipo delle annotazioni da interpretare come istanze dell'algoritmo. Per quanto riguarda *sequenceSpan*, si tratta di un parametro opzionale che assume però rilevanza per algoritmi di *sequence tagging*, che classificano le istanze all'interno di una determinata sequenza analizzando le altre che ne fanno parte; la dimensione di tale sequenza è generalmente rappresentata dalle entità di tipo Sentence, come mostrato nell'esempio. L'impostazione di *trainingAlgorithm* infine permette di scegliere quale algoritmo utilizzare per il task da eseguire; il comportamento dell'algoritmo è personalizzabile

attraverso la specificazione di una valida opzione del parametro facoltativo *algorithmParameters*.

L'applicazione definita dalla sequenza di risorse descritta è stata dunque eseguita sul *training* corpus, risultando nella creazione dei file che costituiscono il modello "appreso" dall'algoritmo di Learning Framework.

```
<ML-CONFIG>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>category</FEATURE>
<DATATYPE>nominal</DATATYPE>
</ATTRIBUTE>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>kind</FEATURE>
<DATATYPE>nominal</DATATYPE>
</ATTRIBUTE>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>length</FEATURE>
<DATATYPE>numeric</DATATYPE>
</ATTRIBUTE>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>orth</FEATURE>
<DATATYPE>nominal</DATATYPE>
</ATTRIBUTE>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>string</FEATURE>
<DATATYPE>nominal</DATATYPE>
</ATTRIBUTE>

</ML-CONFIG>
```

Figura 8. Contenuto del file di configurazione delle feature

5.6.4 Fase di test

L'applicazione utilizzata nella fase di test risulta molto simile a quella di *training*, con però alcune differenze. Mentre le PR di ANNIE utilizzate

precedentemente continuano a essere necessarie per fornire all'algoritmo di *chunking* gli stessi attributi su cui si è basata la creazione del modello di estrazione delle entità, non è da eseguire la risorsa di Annotation Set Transfer, in quanto questa volta non si vuole che l'algoritmo di ML riceva in input le annotazioni relative alle entità. Inoltre la Training Chunking PR di Learning Framework dovrà essere ovviamente sostituita con la corrispondente risorsa applicativa, LF_ApplyChunking. I parametri da impostare in questa PR sono meno, in quanto il comportamento dell'algoritmo da eseguire è in gran parte definito dal modello ottenuto nella fase di *training*; bisognerà però specificare dove la risorsa dovrà reperire tale modello, riportando nel parametro *dataDictionary* lo stesso URL inserito precedentemente. Inoltre anche i parametri *instanceType* e *sequenceSpan* dovranno essere impostati rispecchiando la configurazione utilizzata in fase di *training.* In più, a livello applicativo, è possibile impostare i parametri *outputAsName*, per definire il nome dell'*annotation set* contenente le annotazioni restituite, e *confidenceThreshold*, che permette di indicare una soglia di sicurezza sotto la quale il sistema deve ignorare l'entità individuata.

Una volta definita in questo modo, l'applicazione può essere eseguita fornendo nuove annotazioni di entità del tipo specificato in *classAnnotationType* in fase di *training*.

La Figura 9 mostra il risultato dell'applicazione dell'algoritmo di *chunking* su un documento del corpus di test per l'individuazione di entità di tipo Person.

Figura 9. Risultato dell'estrazione di entità di tipo Person effettuata con Learning Framework

5.6.5 Analisi e valutazione dei risultati

GATE Developer fornisce diverse funzionalità per l'analisi delle annotazioni risultanti dall'esecuzione di un'applicazione, utili per valutare le performance di un sistema sviluppato.

Un primo processo di analisi che si può fare è un confronto diretto all'interno di un documento tra annotazioni dello stesso tipo appartenenti a set diversi, generati in modo diverso. Questo può essere fatto attraverso la modalità di visualizzazione Annotation Stack, che su una stessa porzione di testo in cui è presente un'annotazione in almeno uno dei due set, mostra le entità annotate di uno e dell'altro, in modo da porle in confronto diretto. La Figura 10 mostra due esempi di visualizzazione delle annotazioni in modalità Annotation Stack, uno in cui c'è una corrispondenza delle entità annotate e una in cui la porzione di testo è rappresentata in modo diverso dai due set.

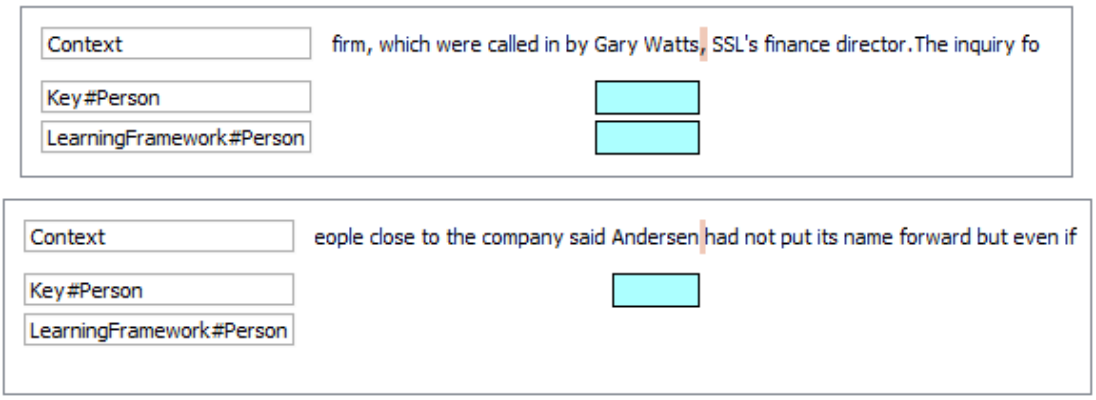


Figura 10. Confronto di entità annotate tramite Annotation Stack

Un altro strumento estremamente utile nella valutazione dei risultati ottenuti da un'applicazione in GATE è il Corpus Quality Assurance. Questo *tool* è utilizzabile su un corpus, all'interno del *resource viewer*, e permette di calcolare le misure di *precision*, *recall* e *F-score* per ogni annotazione associata ai documenti che lo compongono. Attraverso una serie di comandi forniti è possibile indicare uno o più tipi di annotazione da valutare, la versione di F-score che si vuole calcolare (variabile in base al coefficiente β e alla rigidità di assegnazione di un *match* valido) e la selezione di due *annotation set,* uno etichettato come A, che rappresenta il *gold standard,* e l'altro B di annotazioni da confrontare; in base ai parametri forniti, verranno visualizzati i tre valori per ogni annotazione selezionata. Un esempio di utilizzo di Corpus Quality Assurance è mostrato in Figura 11.

Lo strumento è stato usato per valutare le diverse performance ottenute eseguendo tre diversi test relativi all'estrazione di diversi tipi di entità. I risultati sono mostrati nella Tabella 3.

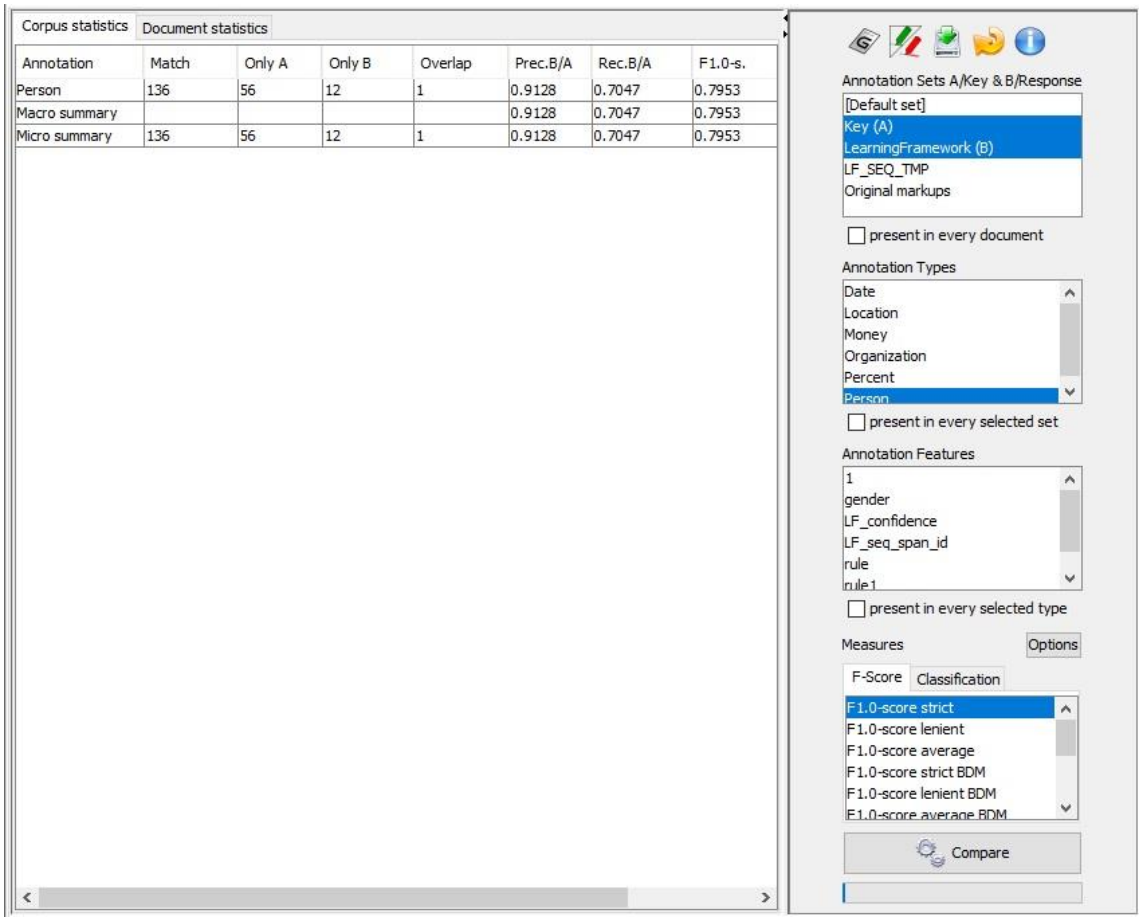


Figura 11. Utilizzo del tool Corpus Quality Assurance

Entità	***Precision***	***Recall***	***F1-score strict***
PERSON	0.9128	0.7047	0.7953
ORGANIZATION	0.7550	0.6914	0.7218
LOCATION	0.9238	0.6667	0.7745

Tabella 3. Valutazione dei risultati ottenuti nei test sull'estrazione di entità con Learning Framework

6. Conclusioni

Nel corso del testo sono stati analizzati diversi aspetti di *Natural Language Processing* e delle tematiche correlate. Inizialmente si è cercato di individuare le principali problematiche di ricerca sul tema dell'elaborazione del linguaggio naturale. In seguito ci si è soffermati sui task di *Information Extraction* e in particolare sull'attività di Named Entity Recognition, uno dei principali focus della ricerca nel campo di NLP. Il tema è poi stato analizzato focalizzandosi sull'approccio *machine learning*, mostrandone vantaggi e possibili declinazioni. Infine si è dato spazio a una panoramica sul sistema GATE, una delle principali piattaforme per l'elaborazione del linguaggio naturale, descrivendone la struttura e le potenzialità, esemplificate per mezzo di un esempio illustrato passo per passo dello svolgimento di un'attività di NER con algoritmi di ML.

L'analisi effettuata sull'argomento nel corso della stesura del testo ha evidenziato come le sfide e le possibili applicazioni legate all'NLP possano rappresentare un *trend* dei prossimi decenni dalle grandi potenzialità. Questo è motivato da un lato dal continuo aumento di contenuti testuali in linguaggio umano, dovuto in gran parte all'esplosione dei *social network* e ad una pervasività sempre maggiore della tecnologia nella società, dall'altro da un interesse sempre più evidente verso tecnologie di intelligenza artificiale, che rendono necessari meccanismi di comunicazione diretta tra uomo e macchina. Inoltre lo sviluppo di tecniche sempre più avanzate, in particolar modo di tipo statistico *unsupervised* o *semisupervised,* coadiuvato dalla disponibilità di sistemi computazionali sempre più performanti, potrebbero mettere le basi per progressi importanti in tale ambito.

Alcune sfide interessanti potrebbero essere legate all'approfondimento

dell’analisi semantica e pragmatica del linguaggio umano, che restringerebbe la distanza che ancora c’è tra il modo di interpretare la realtà da parte dei sistemi informatici e dell’uomo; una distanza che rende ancora difficile, ad esempio, creare un web semantico che possa permettere realmente di accedere a conoscenze mirate, nei contesti più disparati, in modo semplice e naturale.

Bibliografia

ALPAYDIN, E. (2010). ***Introduction to Machine Learning, Second Edition*** (pp. 1-14). The MIT Press.

BERNERS-LEE, T., BIZER, C., HEATH, T. (2009). **Linked Data - The Story So Far**. In T. Heath, M. Hepp, C. Bizer. *International Journal on Semantic Web and Information Systems*. IGI Global.

CAMBRIA, E., WHITE, B. (2014). ***Jumping NLP Curves: A Review of Natural Language Processing Research***, IEEE Computational Intelligence Magazine, May 2014.

COWIE, J., WILKS, Y. (1996). ***Information Extraction***.

CUNNINGHAM, H. (2014). ***Developing Language Processing Components with GATE Version 8***. University of Sheffield Department of Computer Science.

DALE, R. (2010). **Classical Approaches to Natural Language Processing**. In N. Indurkhya and F. J. Damerau, *Handbook of Natural Language Processing, 2nd ed.* (pp. 3-7). CRC Press.

ERDOGAN, H. (2010). ***Sequence labeling: generative and discriminative approaches***. ICMLA 2010 Tutorial, Bethesda, MD

HOBBS, J. R., RILOFF, E. (2010). **Information Extraction**. In F. J. Nitin Indurkhya, *Handbook of Natural Language Processing, 2nd ed.* (p. 523). CRC Press.

HUTCHINS, W. J., SOMERS, H. L. (1992). **General introduction and brief history**. In *An introduction to machine translation* (pp. 1-9). London: Academic Press.

JOHNOSON, M. (2009). ***How the statistical revolution changes (computational) linguistics.*** Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics.

JURAFSKY, D., MARTIN, J. H. (2009). ***Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition. 2nd Edition*** (pp. 1-19, 575-581). Prentice-Hall.

MITCHELL, T. (1997). ***Machine Learning*** (pp. 1-4). McGraw-Hill.

PALMER, D. D. (2010). **Text Preprocessing**. In N. Indurkhya and F. J. Damerau, *Handbook of Natural Language Processing, 2nd ed.* (pp. 9-30). CRC Press.

PISKORSKI, J., YANGARBER R. (2012). **Information Extraction: Past, Present and Future**. In AA.VV., *Multi-source, Multilingual Information Extraction and Summarization* (pp. 23-49). Springer.